JavaScript

Part #1

History

- Originally developed by Branden Eich (Netscape), as LiveScript
- Became a joint venture of Netscape and Sun in 1995, renamed JavaScript
- Now standardized by the European Computer Manufacturers Association as ECMA-262 (also ISO 16262)
- JavaScript and Java are only related through syntax
  - JavaScript is dynamically typed
  - JavaScript's support for objects is very different
- JavaScript for Web platform
  - Speed
  - Gadgets
  - Mashups

JavaScript Features - Crockford

- Load and go delivery
- Case sensitive
- Loose typing (or dynamic typing)
- Objects as general containers
  - root object is Object
  - add properties to object, clone objects
  - objects are accessed through references
- Inheritance
  - extends keyword
  - Prototypal
- Lambda

What tools you need to learn JavaScript

- Same as HTML and CSS
  - At least for the moment
- Text editor
- Web browser
- No need Web server

General Syntax

- Import a JavaScript file

```
<script type = "text/javaScript"
          src = "myScript.js">
</script>
```

- Embed JavaScript code

```
<script type = "text/javaScript">

   //-- JavaScript script –

</script>
```

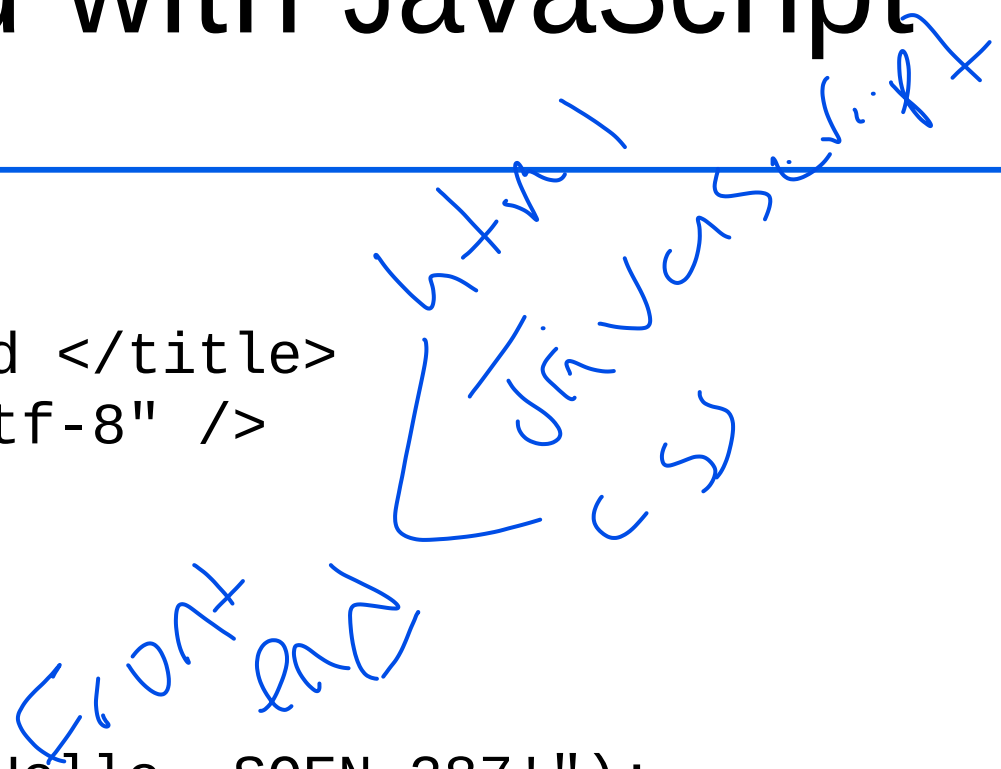
- JavaScript comments: both `//` and `/* … */`

Hello World with JavaScript

```
<html lang = "en">
  <head>
    <title> Hello world </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <script>

      document.write("Hello, SOEN 287!");

    </script>
  </body>
</html>
```



Operations

- Numeric operators `++`, `--`, `+`, `-`, `*`, `/`, `%`
- The `Math` Object provides `floor`, `round`, `max`, `min`, trig functions, etc.
  - e.g., `Math.cos(x)`

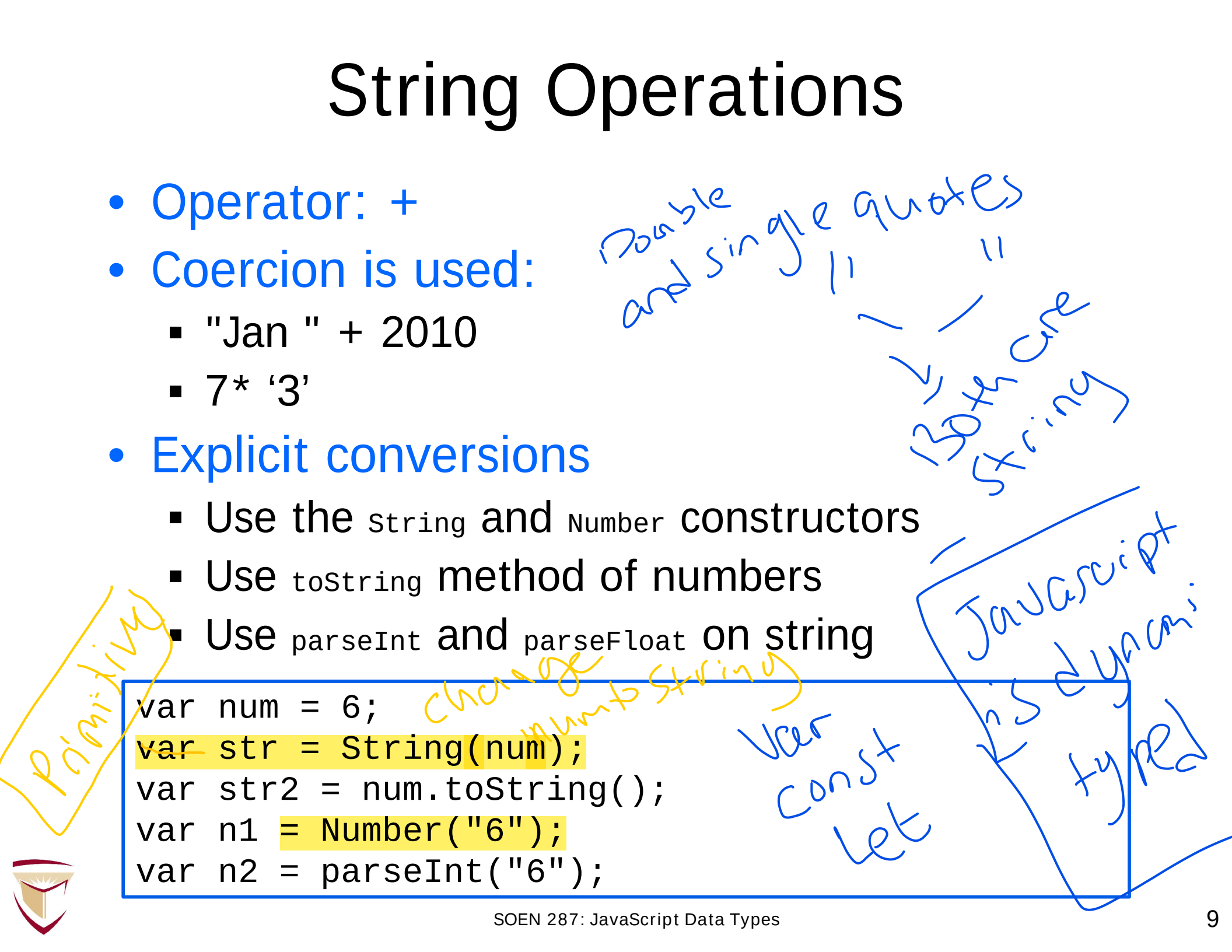
The `Number` Object

- `MAX_VALUE`, `MIN_VALUE`, `NaN`, `POSITIVE_INFINITY`, `NEGATIVE_INFINITY`, `PI`
- e.g., `Number.MAX_VALUE`
- An arithmetic operation that creates overflow returns `NaN`
- `NaN` is not `==` to any number, not even itself
- Test for it with `isNaN(x)`
- `Number` object has the method, `toString`

String Operations

- Operator: +
- Coercion is used:
  - "Jan " + 2010
  - 7* '3'
- Explicit conversions
  - Use the `String` and `Number` constructors
  - Use `toString` method of numbers
  - Use `parseInt` and `parseFloat` on string

```
var num = 6;
var str = String(num);
var str2 = num.toString();
var n1 = Number("6");
var n2 = parseInt("6");
```



+: both plus and string concatenation – be careful

```
1 + 2
"1" + 2
1 + "2"
"1" + "2"
1+ " bird"
1+2+ "birds"
```

+: both plus and string concatenation

```
1 + 2
"1" + 2
1 + "2"
"1" + "2"
1+ " bird"
1+2+ "birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

+: both plus and string concatenation

```
1 + 2  = 3
"1" + 2
1 + "2"
"1" + "2"
1+ " bird"
1+2+ "birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

+: both plus and string concatenation

```
1 + 2  = 3
"1" + 2 = "12"
1 + "2"
"1" + "2"
1+ " bird"
1+2+ "birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

+: both plus and string concatenation

```
1 + 2  = 3
"1" + 2 = "12"
1 + "2" = "12"
"1" + "2"
1+ " bird"
1+2+ "birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

+: both plus and string concatenation

```
1 + 2  = 3
"1" + 2 = "12"
1 + "2" = "12"
"1" + "2" = "12"
1+ " bird"
1+2+ "birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

+: both plus and string concatenation

```
1 + 2  = 3
"1" + 2 = "12"
1 + "2" = "12"
"1" + "2" = "12"
1+ " bird" = "1 bird"
1+2+ "birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

+: both plus and string concatenation

```
1 + 2  = 3
"1" + 2 = "12"
1 + "2" = "12"
"1" + "2" = "12"
1+ " bird" = "1 bird"
1+2+ "birds" = "3birds"
```

- If both operands are numbers:
  - + is addition
- otherwise string concatenation.

Question

- What is the result of `'$' + 3 + 4` ?

A. $7

B. $34

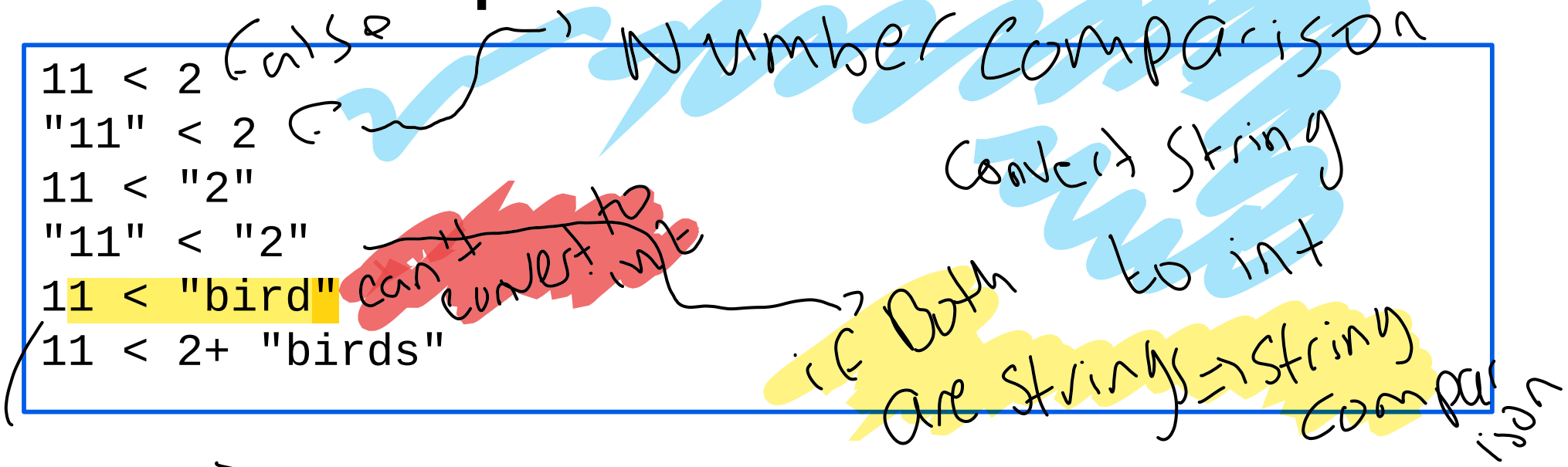
C. error

D. undefined

Other operators in this case?

```
11 < 2
"11" < 2
11 < "2"
"11" < "2"
11 < "bird"
11 < 2+ "birds"
```

Other operators in this case?



- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Other operators in this case?

```
11 < 2                    false
"11" < 2
11 < "2"
"11" < "2"
11 < "bird"
11 < 2+ "birds"
```

- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Other operators in this case?

```
11 < 2                  false
"11" < 2                false
11 < "2"
"11" < "2"
11 < "bird"
11 < 2+ "birds"
```

- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Other operators in this case?

```
11 < 2                  false
"11" < 2                false
11 < "2"                false
"11" < "2"
11 < "bird"
11 < 2+ "birds"
```

- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Other operators in this case?

```
11 < 2                  false
"11" < 2                false
11 < "2"                false
"11" < "2"              true
11 < "bird"
11 < 2+ "birds"
```

- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Other operators in this case?

```
11 < 2                  false
"11" < 2                false
11 < "2"                false
"11" < "2"              true
11 < "bird"             false
11 < 2+ "birds"
```

- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Other operators in this case?

```
11 < 2                  false
"11" < 2                false
11 < "2"                false
"11" < "2"              true
11 < "bird"             false
11 < 2+ "birds"         false
```

- If one operand is a number, and the other can be converted to a number, < is a number comparison,
- If one operand is a number, and the other cannot be converted to a number, false all the time.
- If the two operands are string, < is a string comparison

Question

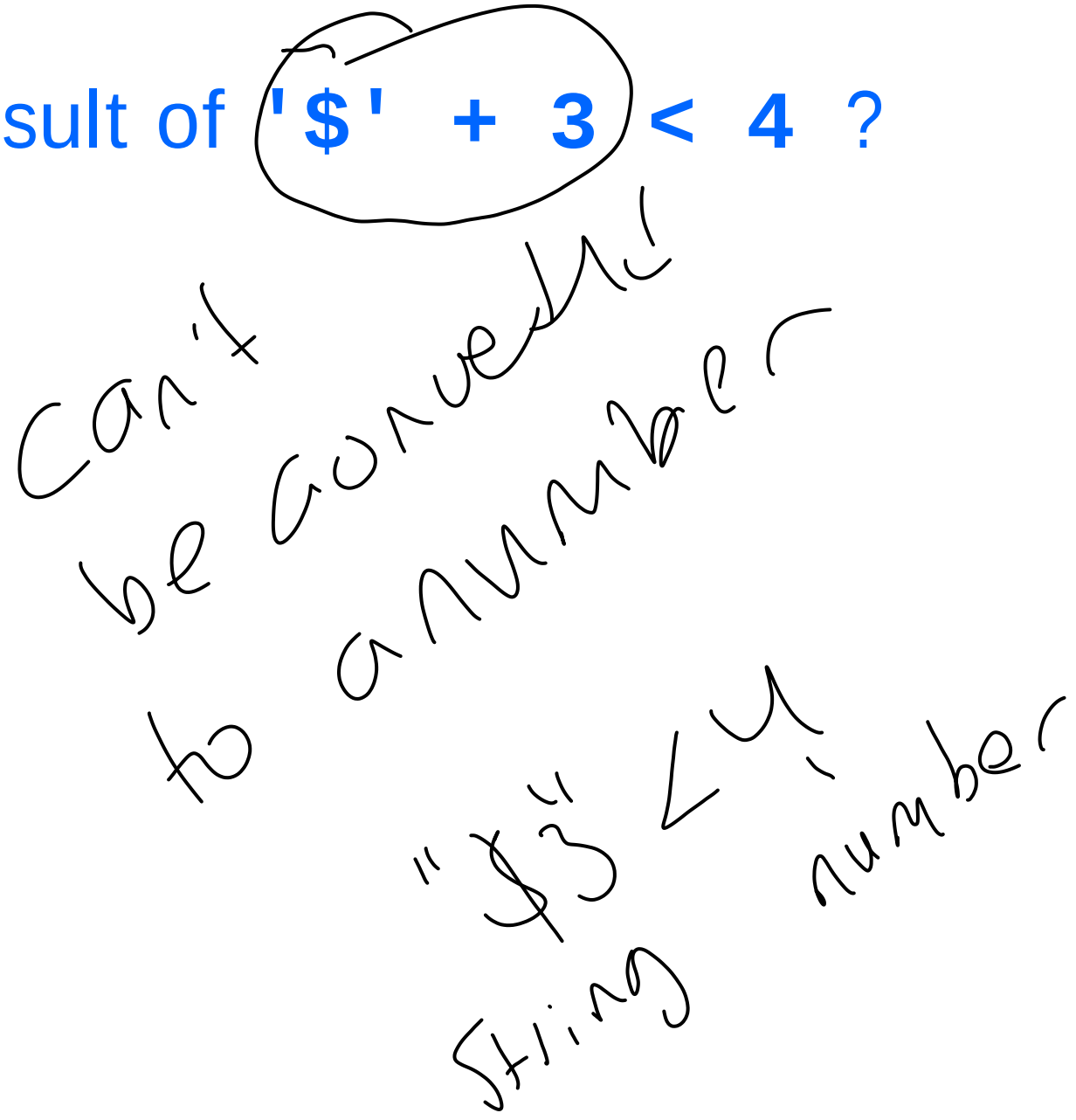
- What is the result of `'$' + 3 < 4` ?

A. false

B. true

C. error

D. undefined



Other operators in this case?

```
11 * 2
"11" * 2
11 * "2"
"11" * "2"
11 * "2bird"
```

22
22
22
22
NaN

because "bird" can't be converted to a number

is NaN to check if it is number or not

Other operators in this case?

```
11 * 2
"11" * 2
11 * "2"
"11" * "2"
11 * "2bird"
```

- If operands are numbers, or all can be converted to a number, * is a number multiply
- If one operand is a number, and the other cannot be converted to a number, NaN all the time.

Other operators in this case?

```
11 * 2                  22
"11" * 2
11 * "2"
"11" * "2"
11 * "2bird"
```

- If operands are numbers, or all can be converted to a number, * is a number multiply
- If one operand is a number, and the other cannot be converted to a number, NaN all the time.

Other operators in this case?

```
11 * 2                  22
"11" * 2                22
11 * "2"
"11" * "2"
11 * "2bird"
```

- If operands are numbers, or all can be converted to a number, * is a number multiply
- If one operand is a number, and the other cannot be converted to a number, NaN all the time.

Other operators in this case?

```
11 * 2                  22
"11" * 2                22
11 * "2"                22
"11" * "2"
11 * "2bird"
```

- If operands are numbers, or all can be converted to a number, * is a number multiply
- If one operand is a number, and the other cannot be converted to a number, NaN all the time.

Other operators in this case?

```
11 * 2                  22
"11" * 2                22
11 * "2"                22
"11" * "2"              22
11 * "2bird"
```

- If operands are numbers, or all can be converted to a number, * is a number multiply
- If one operand is a number, and the other cannot be converted to a number, NaN all the time.

Other operators in this case?

```
11 * 2                  22
"11" * 2                22
11 * "2"                22
"11" * "2"              22
11 * "2bird"            NaN
```

- If operands are numbers, or all can be converted to a number, * is a number multiply
- If one operand is a number, and the other cannot be converted to a number, NaN all the time.

Question

- What is the result of `'$'*3 < 4` ?

A. false

B. true

C. error

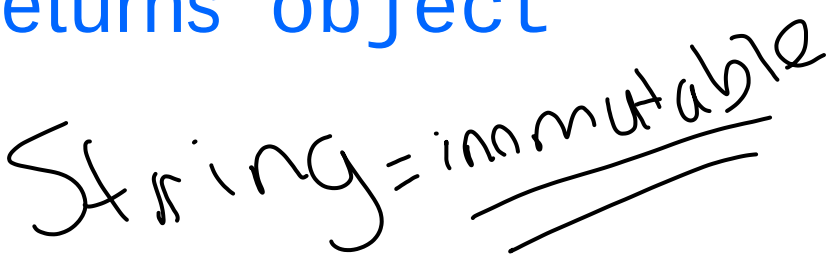
D. undefined

'$'*3

$ can't be converted

`string`

- Sequence of 0 or more 16-bit characters
- No separate character type
  - Characters are represented as strings with a length of 1
- Strings are immutable (similar to Java!)
- Use `==` to check if the values of the strings are the same (definitely not in Java!)
- String literals can use single or double quotes
- `String.length`
- `String(value)`: returns `string`
- `new String(value)`: returns `object`
  - You can live without this



Question

- The following code prints

```
var a = "123";
var b = "123";
document.write(a==b);
```

A. true
B. false
C. error
D. undefined

String methods

- charAt
- concat
- indexOf
- lastIndexOf
- match
- replace
- search
- slice
- split
- substring
- toLowerCase
- toUpperCase

Boolean

- Boolean values are `true` and `false`
- 0, -0, null, "", false, undefined, or NaN are considered **`false`**
- "0" is **`true!`**
- the `Boolean(value)` function

Question

- What does the following code return?

```
Boolean("false");
```

A. true
B. false
C. error
D. undefined

The `Date` Object

- The `Date` Object

`toLocaleString` – returns a string of the date

`getDate` – returns the day of the month

`getMonth` – returns the month of the year (0 – 11)

`getDay` – returns the day of the week (0 – 6)

`getFullYear` – returns the year

`getTime` – returns the number of milliseconds since January 1, 1970

`getHours` – returns the hour (0 – 23)

`getMinutes` – returns the minutes (0 – 59)

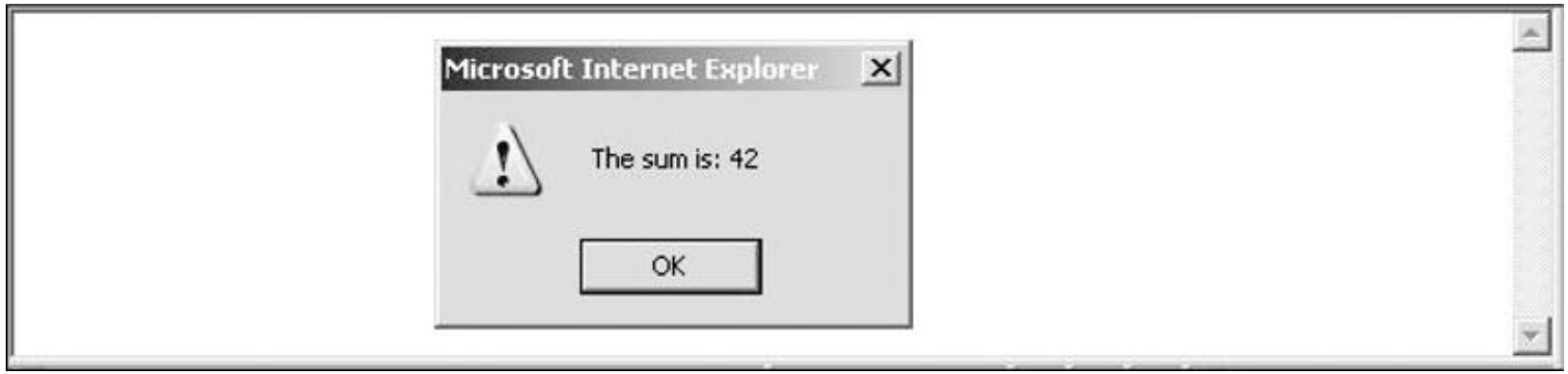
`getMilliseconds` – returns the millisecond (0 – 999)

Screen Output & Keyboard Input

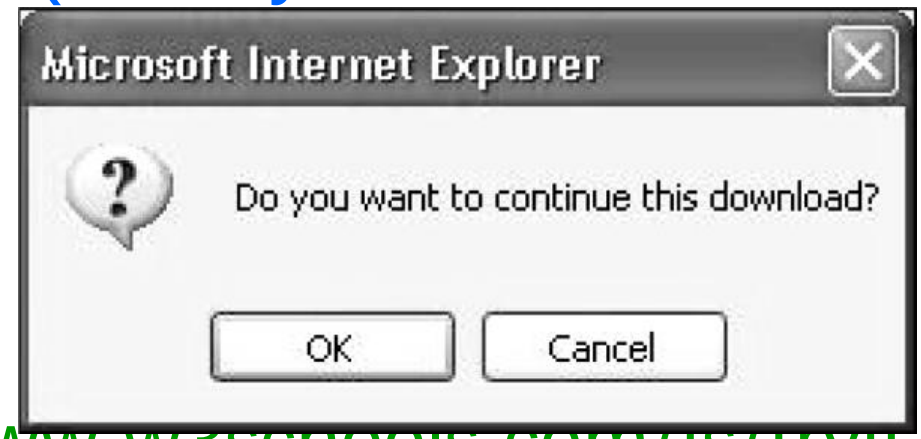
- The model for the browser display window is the `Window` object
- The `Window` object contains `document` object
- The `Document` object has a method, `write`, which dynamically creates content

Screen output

- `alert("The sum is:"+sum+"\n");`



- http://www.w3schools.com/js/tryit.asp?filename=tryjs_alert
- `confirm("Do you want to continue?");`



- http://www.w3schools.com/js/tryit.asp?filename=tryjs_confirm

Get input in a dialog box

- `prompt("What is your name?", "");`

Hoisting

variable is visible to the things above it

Control expressions

```
if(1) {document.write('yes');}
      else {document.write('no');}
if(0) {document.write('yes');}
      else {document.write('no');}
```

- 0, -0, null, "", false, undefined, or NaN are considered **`false`**
- `==, !=, <, >, <=, >=, ===, !==`
- `&&, ||, !, ! !`

the value is visible but the variable is not

Equal and not equal

- `==` and `!=` can do type coercion
- `===` and `!==` cannot do type coercion
- Thus

```
"3" == 3: true
"3" === 3: false
```

Question

```
var a = "123";
var b = "123";
if(a==b) {document.write('yes');}
      else {document.write('no');}
```

- What is the output?

A. yes

B. no

C. error

D. nothing

Question

```
if(3!== "3") {document.write('yes');}
      else {document.write('no');}
```

- What is the output?

A. yes

B. no

C. error

D. nothing

Question

```
var a = new String("123");
var b = new String("123");
if(a==b) {document.write('yes');}
      else {document.write('no');}
```

- What is the output?

A. yes

B. no

C. error

D. nothing

A Challenge Question

```
var a = String("123");
var b = new String("123");
if(a==b) {document.write('yes');}
      else {document.write('no');}
```

- What is the output?

  A. yes

  B. no

  C. error

  D. nothing

The logic operators: && and ||

- &&:

  if the first operand is truthy,

  return the second operand,

  else return the first operand

- ||:

  if the first operand is truthy,

  return the first operand,

  else return the second operand

```
var last = input || default_value;
```

The logic operators: !

- `!:`

  if the operand is truthy,

  return `false`,

  else return `true`

- `!!:` as `Boolean(value)`, return `false` or `true`.

The bitwise operators:

Operator	Description	Example	Same as	Result	Value
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

http://www.w3schools.com/jsref/jsref_operators.asp

Control Statements

- Switch

```
switch (expression) {
     case value_1:
        // value_1 statements
     case value_2:
        // value_2 statements
     …
     [default:
        // default statements]
  }
```

Control Statements

- Loop
  - `while`
  - `for`
  - `do-while`